Lorikeet: An Efficient Multicast Protocol for the Distribution of Multimedia Streams

Justin Viiret

Thesis submitted for the degree of

Doctor of Philosophy

in

Electrical and Electronic Engineering

at

The University of Adelaide

(Faculty of Engineering, Computer and Mathematical Sciences)

School of Electrical and Electronic Engineering

June 4, 2007

Contents

Abstract

Internet Protocol multicast has been standardised since the late 1980's, but is yet to be extensively deployed by most Internet Service Providers. Many organisations are not willing to bear the additional router CPU load and memory requirements that multicast entails, and the IP multicast suite of protocols requires deployment on every router spanned by the multicast group to operate. Additionally, these protocols are predominantly designed for the general case of multiple-source, multiple-receiver transmission and can be complex and inefficient to use in simpler scenarios.

Single-source streaming of multimedia on the Internet is rapidly becoming a very popular application, and is predominantly being served by content providers using simultaneous unicast streams. A multicast transmission protocol designed for this application that can operate without requiring a widely deployed IP multicast infrastructure has the potential to save content-providers and network service providers significant amounts of bandwidth. This protocol should provide packet duplication and forwarding capabilities on routers in the network, rather than pushing this functionality to the receivers themselves, requiring them to become part of the multicast infrastructure.

We describe Lorikeet, a new protocol for the multicast distribution of multimedia streams from a single source. This protocol builds its multicast tree from the source, discovering routers that support the protocol in the network and using them to provide branching in the tree. The tree itself is managed in a decentralised fashion, with joining receivers finding parent routers through a limited, recursive search of the tree. On a participating node, information about the tree's structure is limited

to the addresses of that node's children and its path through the tree back to the source. Unlike most other multicast protocols, a new receiver is connected to the tree using its forward path from the source and packets are delivered through the tree via hop-by-hop delivery over unicast connections between nodes. Lorikeet also actively maintains the tree structure using a localised rearrangement algorithm triggered by a topological change in the tree structure. This rearrangement allows the tree to remain efficient in the face of changes to the receiver population, which can change the shape of the tree over time.

Lorikeet is designed to operate with no further protocol support than that provided by existing Internet unicast protocols. It requires none of the standard IP multicast infrastructure, such as Class D group addressing. Its use of unicast connections between nodes allows it to be deployed incrementally on the network, and its behaviour will degrade to simultaneous unicast when no routers that support the protocol are present at all. However, significant performance gains can be achieved even when there are only a few supporting routers present in the network: Lorikeet produces trees with half the cost of a unicast tree when just 10% of routers are Lorikeet-capable. Lorikeet's tree construction and rearrangement algorithms generate multicast trees of comparable total cost to those created by algorithms of considerably higher message complexity, such as those that employ exhaustive searches of the tree during joins.

We develop the Lorikeet protocol from a set of requirements based on its target application and the properties of the current Internet. After describing the protocol's behaviour, we analyse its message complexity and its performance in terms of tree cost. We also analyse several other multicast protocols from the research literature, comparing their performance to that of Lorikeet in both complete deployment and incremental deployment scenarios.